UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/691,414	10/22/2003	Andrew Nuss	11103	6165

27015 7590 04/07/2005

CHARLES LOUIS THOEMING
1390 WILLOW PASS ROAD, SUITE 1020
CONCORD, CA 94520

EXAMINER	
FOWLKES, ANDRE R	
ART UNIT	**PAPER NUMBER**
2192	

DATE MAILED: 04/07/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/691,414	NUSS, ANDREW	
	Examiner	Art Unit	
	Andre R. Fowlkes	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *3/24/05*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-57* is/are pending in the application.

4a) Of the above claim(s) *44 and 47* is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-43,45,46 and 48-57* is/are rejected.

7)☒ Claim(s) *44 and 47* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-57 are pending.

Specification

2. The abstract of the disclosure is objected to because the abstract exceeds 150 words in length. Correction is required. See 35 U.S.C. 111 and MPEP § 608.01(b).

3. The use of the trademark Java™ has been noted in this application. It should be capitalized wherever it appears and be accompanied by the generic terminology.

Although the use of trademarks is permissible in patent applications, the proprietary nature of the marks should be respected and every effort made to prevent their use in any manner which might adversely affect their validity as trademarks.

4. Upon review of the specification, it appears possible that numerous inventions were disclosed. Applicant's cooperation is requested in clearly disclosing and claiming the one invention to be prosecuted in the instant application.

Claim Objections

5. Claims 44 and 47 are objected to under 37 CFR 1.75(c) as being in improper form because a multiple dependent claim cannot serve as a basis for any other multiple dependant claim. See MPEP § 608.01(n). Accordingly, claims 44 and 47 have not been further treated on the merits.

Claim Rejections - 35 USC § 101

6. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

7. Claims 1-57 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

8. The invention, as disclosed in claims 1 and 2, is directed to non-statutory subject matter. While the claims is in the technological arts, it is only a listing without any interrelationship and is therefore is directed to nonfunctional descriptive material which is not performing any useful act and also is not a method, machine, manufacture or composition of matter. Additionally, the claims are not tangibly embodied in a manner so as to be executable. See MPEP 2106(IV)(B)(1)(b).

9. On this basis, claims 3-57, which depend from claims 1 or 2, are also rejected under 35 U.S.C. 101 for containing only a listing of features that are not interrelated and do not remedy the deficiencies of claims 1 or 2.

Claim Rejections - 35 USC § 102

10. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

11. Claims 1-43, 45-46 and 48-57 are rejected under 35 U.S.C. 102(b) as being anticipated by Friedl, "Mastering regular expressions", 2nd edition, ISBN: 0-596-00289-0.

As per claim 1, Friedl discloses **a novel programming grammar for building regular expressions** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data").

As per claim 2, Friedl discloses **a novel programming grammar for building regular expressions which models C-style languages in terms of statement blocks, function blocks, scoped variables and their declarations, expression binding rules, primitive data types, and array declarations** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular

expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++ (i.e. C-style languages in terms of statement blocks, function blocks, scoped variables and their declarations, expression binding rules, primitive data types, and array declarations), offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience").

As per claim 3, the rejection of claim 2 is incorporated and further, Friedl discloses that **the composition of a regular expression uses the standard expression operators and expression binding rules of the C-language such that a regular expression consists of compositions using unary and binary operators of the C-language following exactly the C-model of associativity and precedence for borrowed/overloaded operators, comprising literals and variables as with C-expressions, and further comprising new operators unfound in C-like languages but which follow the C-rules of associativity and precedence, and allowing for the novel regular expression forms of this grammar** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language,

allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++ (i.e. C-style languages in terms of statement blocks, function blocks, scoped variables and their declarations, expression binding rules, primitive data types, and array declarations), offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience").

As per claim 4, the rejection of claim 3 is incorporated and further, Friedl discloses that **the grammar further comprises C/C++ expression syntax for building regular expressions wherein existing C-style unary and binary operators are overloaded to form certain regular expression compositions, wherein existing binary operators are used as unary operators, and wherein new operators are created, in all cases observing the C-language rules of associativity and precedence of operators** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic,

C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading). It appears that applicant is disclosing using operator overloading to perform/compose several different types of regular expression operations. Applicant's cooperation is requested in clearly disclosing and claiming the novel invention to be prosecuted in the instant application.

As per claim 5, the rejection of claim 3 is incorporated and further, Friedl discloses that **the grammar further comprises a novel primitive data type of the language called Pattern, an immutable Object in the same sense as a Java String, which serves to hold ANY regular expression form available in the grammar** (p. 33:1-4, " many programmers independently developed Java regex packages of varying degrees of quality, functionality, and complexity. With the early-2002 release of Java 1.4, Sun entered the fray with their java.util.regex package").

As per claim 6, the rejection of claim 3 is incorporated and further, Friedl discloses that **a union composition regular expression adopts the standard C++ ".vertline." (vertical bar) operator in the normal position in the precedence table of operators, and carries the normal left-to-right associativity for binding in the absence of parentheses** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general

pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 7, the rejection of claim 3 is incorporated and further, Friedl discloses that **a normal (greedy from left to right) concatenation regular expression composition uses the standard C++ "+" (plus sign) operator, chosen to emulate Java string concatenation which is also accomplished by the + operator**(p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 8, the rejection of claim 3 is incorporated and further, Friedl discloses that **a special (greedy from right-to-left) regular expression form of concatenation regular expression composition uses the invented operator "<+" (less than plus sign) in the C++ precedence table at the same position as the + operator and wherein the associativity of concatenation and the choice of right-to-left or left-to-right operators are significant** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 9, the rejection of claim 3 is incorporated and further, Friedl discloses that **a normal (greedy from left-to-right) regular expression concatenation also uses the invented operator ">+" (greater than plus sign) in the C++ precedence table at the same position as the + operator and with exactly the same meaning and usage as the + operator** (p. 4:14-17, " Regular expressions are

the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 10, the rejection of claim 3 is incorporated and further, Friedl discloses that **the C++ binary "*" (asterisk) operator is used to create regular expressions of the composition semantics "repeat a given regex N or more times", wherein the composition is of the form--Pattern * integer--and wherein this operator binds at the normal position of the * operator in the C precedence table** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's

a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 11, the rejection of claim 3 is incorporated and further, Friedl discloses that **the C++ binary "*" (asterisk) operator is used to create regular expressions of the composition semantics "iterate a given regex from N1 to N2 times, inclusive",--as in Pattern * Range--, wherein the Range data type is new built-in data type of the language and is accomplished by the expression--int..int, wherein ".." (dot-dot) is a special operator of this grammar that binds tightest of any operator in the C-style precedence table of this grammar** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 12, the rejection of claim 3 is incorporated and further, Friedl discloses that **the C++ binary "*" (asterisk) operator is used in the unary sense to**

compose regular expressions of the composition semantics "repeat a given regex 0 or more times" as in--* Pattern--, wherein the unary repeat operator must be placed before the pattern in order to fit into the C-expression model (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++ offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 13, the rejection of claim 3 is incorporated and further, Friedl discloses that **the "+" (plus sign) operator is used in the unary sense to compose regular expressions of the composition semantics "repeat a given regex 1 or more times" as in--+Pattern--, wherein the unary repeat operator must be placed before the pattern in order to fit into the C-expression model** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally

fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 14, the rejection of claim 3 is incorporated and further, Friedl discloses that **the "?" (question mark) metacharacter-operator is used to optionalize a regular expression, wherein the optionalize-grammar uses this operator preceding the pattern, and wherein the operator-metacharacter has a unary purpose in addition to its standard binary purpose** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience").

As per claim 15, the rejection of claim 3 is incorporated and further, Friedl discloses that **the conditional regular expression uses the "?" (question mark)**

operator as in--bool ? Pattern1: Pattern2--as a natural by-product of how the C-interpreter handles the conditional operator (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 16, the rejection of claim 1 or 3 is incorporated and further, Friedl discloses that **arbitrary instructional side-effects are embedded directly into regular expressions through the novel do-pattern form and through other novel composition forms which are implemented in whole or in part via implicit do-patterns** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and

C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 17, the rejection of claim 16 is incorporated and further, Friedl discloses that **the syntax of the explicit do-pattern comprises--do (preStmt1, preStmt2, . . . , preStmtN1; embed-regex; postStmt1, postStmt2, . . . , postStmtN2)--which is modeled closely after the C-style for-statement, wherein comma separated statements are embedded before and after the loop continuation expression, "do" is substituted for "for", and the loop-continuation-expression is replaced with a regular expression** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 18, the rejection of claim 3 is incorporated and further, Friedl disclose **a CapturePattern syntax wherein a regular expression has its match results captured into a named variable of any scope available to the CapturePattern, the named variable consisting of a global variable, a function parameter, a production rule parameter, a variable declared in a function block, or a statement block of the function, or a variable declared within the scope of a DoPattern, and wherein the grammar is--&myvar (reg-ex)--** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 19, the rejection of claim 16 is incorporated and further, Friedl discloses that **instantiated production rules generate implicit side-effects for proper binding of the in-out parameters and out parameters of the reference to the actual variable being passed** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a

general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 20, the rejection of claim 16 is incorporated and further, Friedl discloses that **the side effects of any composition can be nullified by the novel inhibit grammar comprising--inhibit (regex)--** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 21, the rejection of claim 1 or 3 is incorporated and further, Friedl discloses that **a subjunctive grammar is used to qualify or reject matches to the primary via the secondary wherein any side-effects embedded in the secondary are inhibited during qualification/rejection, and wherein any side-effects embedded in the primary are not affected by qualification/rejection since if qualification occurs, the expression executes exactly as if the subjunctive expression had been replaced by the primary and if rejection occurs, there is no match and no candidate for a winning thread, hence no instructional side-effects** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 22, the rejection of claim 1 or 3 is incorporated and further, Friedl discloses **means for the programmer to control side-effects actually triggered as a result of matching a regular expression to a stream/string without ambiguity** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text

processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 23, the rejection of claim 22 is incorporated and further, Friedl discloses that **means for the programmer to control side-effects actually triggered as a result of matching a regular expression to a stream/string without ambiguity further comprises instruction-arcs which represent all side-effects in the automata, which do not contribute to character-matching characteristics of any automata thread, and which therefore do not affect the accept/reject outcome of any composition or sub-composition automata** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression

library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 24, the rejection of claim 23 is incorporated and further, Friedl discloses that **the instruction arcs do affect the arcnum-sequence priority given to arcs which helps select a winning automata execution thread to resolve ambiguity** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 25, the rejection of claim 23 is incorporated and further, Friedl discloses that **adding or removing side-effects to/from portions of a regular expression does not affect whether or not any other side-effects will be part of the winning thread** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern

notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 26, the rejection of claim 23 is incorporated and further, Friedl discloses that **if side-effects from a sub-expression of a regular expression are part of a winning thread, than any outer low-level implicit or explicit do-patterns which also contain the side-effects will also have their side-effects included on the winning path, in proper nested order** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 27, the rejection of claim 23 is incorporated and further, Friedl discloses that **upon completion of automata execution and acceptance of the input string, all side-effects triggered for post-acceptance execution reside in one and only one coherent path through the composition automata, providing at most one set of side-effects fired for the execution of a given automata** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 28, the rejection of claim 23 is incorporated and further, Friedl discloses that **all rejected candidate paths through the automata which are rejected at any stage of automata execution thread pruning or during subset construction can only be rejected by the existence of a superior path through the graph that recognizes the same characters according to the ambiguity priority rules of the unary and binary operators of the grammar, and wherein binary rules**

can be described as--P=P1 op P2--. (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 29, the rejection of claim 23 is incorporated and further, Friedl discloses that **in terms of left-to-right concatenation, the left term is to be greedier than the right term wherein if the composition or sub-composition recognizes the same set of characters in multiple allocations to the two terms, the winning automata thread will select the allocation of the fewest characters to the right term, and which is significant only when the concatenation composition contains side-effects** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and

data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 30, the rejection of claim 23 is incorporated and further, Friedl discloses that **in terms of right-to-left concatenation, the right term is to be greedier than the left term wherein if the composition or sub-composition recognizes the same set of characters in multiple allocations to the two terms, the winning automata thread will select the allocation of the fewest characters to the left term, and which is significant only when the concatenation composition contains side-effects** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 31, the rejection of claim 23 is incorporated and further, Friedl discloses that **for union when both terms match exactly the same number of characters, the right term is always preferable to the left term, and which is significant only when the concatenation composition contains side-effects** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 32, the rejection of claim 31 is incorporated and further, Friedl discloses that **for union when the terms match a different number of characters, the greedier term is preferred as a result of higher importance (of greediness) than left-to-right or top-to-bottom precedence in the union** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally

fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 33, the rejection of claim 23 is incorporated and further, Friedl discloses that **for repeat or iterate compositions defined by--P=P1 op N, where N is either the minimum number of iterations expected or the iteration from-to-range, ambiguity resolution rules provide that no losing path through the sub-expression P will match a greater number of characters on its first match to P1, and likewise in turn on each subsequent match, and which is of possible significance in the presence of side-effects within P1** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 34, the rejection of claim 33 is incorporated and further, Friedl discloses that **ambiguity resolution rules are covered for unary repeats in that the unary repeats can be converted to binary repeats** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 35, the rejection of claim 23 is incorporated and further, Friedl discloses that **ambiguity resolution rules proceed from higher composition levels to lower levels wherein the optimal thread is chosen based upon a mismatch between two threads at the highest level of composition and is justified by the ambiguity resolution rules involving sequencing of arcnum sequences, and wherein (by example) inner repeats are as greedy as possible relative to outer repeats (relative to side-effects), and wherein the associativity binding of a concatenation of 3 or more terms can be significant in the presence of side**

effects in any of those terms (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 36, the rejection of claim 23 is incorporated and further, Friedl discloses that **ambiguity resolution rules provide that when matching a stream against a pattern, after all the rules have fired, and two or more paths are presented each of which eats a different number of characters, the path/subpath which eats the greater number of characters is given preference** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared

regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 37, the rejection of claim 1 or 3 is incorporated and further, Friedl discloses that **the production rules are part of the language at any scope where a function can be defined, wherein production rules can be parameterized in the same form and syntax of param-decl-lists of functions, wherein production rules are instantiated (as templates) rather than called, and wherein the syntax for parameterization of a production rule's embedded regular expression is--production MyRule <param-decl-list>reg-ex;--.** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 38, the rejection of claim 37 is incorporated and further, Friedl discloses that **parameterization of a production rule's embedded regular expression allows the recognition characteristics of the instantiated rule to vary through in/default parameters to create reusable design patterns** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 39, the rejection of claim 37 is incorporated and further, Friedl discloses that **parameterization of a production rule's embedded regular expression allows the side-effect characteristics of the instantiated rule to vary through in out, out, or default in container parameters (such as arrays) to create a rule that can be instantiated to capture into or modify parameters rather than actual variables** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text.

With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 40, the rejection of claim 21 is incorporated and further, Friedl discloses that **qualification/rejection of matches by a subjunctive regular expression involves a primary regular expression and a secondary regular expression and an automata composition engine which uses same-time arrival at the exit nodes of both the primary regular expression and the secondary regular expression to determine qualification/rejection** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 41, the rejection of claim 40 is incorporated and further, Friedl discloses that **in the case of the subjunctive ButGrammar a match to the primary regular expression is accepted only if the secondary regular expression also has a same-time arrival to its exit node, wherein the ButGrammar is expressed as--primary-regex but qualifier-regex--, wherein primary-regex and qualifier-regex are any regular expression possible in the language, and wherein instructional side-effects of the secondary/qualifier regular expression are automatically nullified/inhibited, and all instructional side-effects of the primary regular expression are preserved in order, so long as they fall on a winning thread which has not been disqualified by the qualifier regular expression** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 42, the rejection of claim 40 is incorporated and further, Friedl discloses that **in the case of the subjunctive ButnotGrammar a match to the primary regular expression is rejected if the secondary regular expression also has a same-time arrival to its exit node, wherein the ButnotGrammar is expressed as--primary-regex butnot qualifier-regex--, wherein primary-regex and qualifier-regex are any regular expression possible in the language, and wherein instructional side-effects of the secondary/qualifier regular expression are automatically nullified/inhibited and all instructional side-effects of the primary regular expression are preserved in order, so long as they fall on a winning thread which has not been disqualified by the qualifier regular expression** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 43, the rejection of claim 40 is incorporated and further, Friedl discloses that **subjunctive chaining is allowed (with left-to-right associativity), and**

wherein by example the following expression--primary-regex but qualifier-regex1 but qualifier-regex2--is implied, and wherein primary-regex but qualifier-regex1 is itself a primary regular expression and is then subjuncted with qualifier-regex2, as the secondary [qualifier] regular expression (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 45, the rejection of claim 37 is incorporated and further, Friedl discloses that **proper binding (when instantiated) of production rule templates is accomplished by a novel runtime virtual machine abstraction nominated by the author as a template frame** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds

of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 46, the rejection of claim 40 is incorporated and further, Friedl discloses **using the subjunctive to enhance expressivity in lieu of using complex compositions involving negated character-classes** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 48, the rejection of claim 5 is incorporated and further, Friedl discloses that **the method comprising the step of using the pattern data type immutability of the grammar wherein once a Pattern object is instantiated and**

assigned to a variable, it never will change, even if the variable is reassigned (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 49, the rejection of claim 1 or 3 is incorporated and further, Friedl discloses **using the production rules of the grammar which can be parameterized wherein a reusable design pattern need be coded only once as a production rule template, or a few times, as a set of related templates with the same name but different signatures, wherein the reusability of the rule allows parameterization of both the recognition portions of the resulting graph (as with passing another regex as an in parameterization) as well as parameterization of the side-effects through out and in out params or with an in parameterization consisting of a container structure such as an array** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to

describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 50, the rejection of claim 3 is incorporated and further, Friedl discloses that **the method comprising the step of using the reject literal of the grammar to compose dynamic unions wherein the elements of the union are pulled from container objects, such as arrays and the reject provides the union a valid initial value** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 51, the rejection of claim 1 is incorporated and further, Friedl discloses that **the grammar further comprises parameterizable production rules wherein the programmer can abstract both recognition and side-effect characteristics of a useful regular expression** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 52, the rejection of claim 3 is incorporated and further, Friedl discloses that **the grammar further comprises parameterizable production rules wherein the programmer can abstract both recognition and side-effect characteristics of a useful regular expression** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold,

spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 53, the rejection of claim 1 is incorporated and further, Friedl discloses that **the grammar further comprises a design pattern for extracting complex structures via nested do-patterns and capture patterns** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 54, the rejection of claim 3 is incorporated and further, Friedl discloses that **the grammar further comprises a design pattern for extracting complex structures via nested do-patterns and capture patterns** (p. 4:14-17, "

Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 55, the rejection of claim 35 is incorporated and further, Friedl discloses that **a production rule can bind an in-parameterization to itself during instantiation, allowing the rule-body to include do-patterns which manipulate the in-parameterization, such that side-effect statements which reference the in-parameter (within the body-expression of the production rule) are referencing the value at time of instantiation rather than the value at the time of tokenize-statement, a type of "unpinning" which fits well within the goal of step-by-step composition** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and

data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 56, the rejection of claim 1 is incorporated and further, Friedl discloses that **the grammar further comprises a design pattern making a Pattern/regular expression an "in" parameterization of rule, wherein other regular expression instantiations are passed during instantiation yielding a production rule as a formula for building a regular expression (from other regular expressions) rather than the regular expression itself** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 57, the rejection of claim 3 is incorporated and further, Friedl discloses that **the grammar further comprises a design pattern making a Pattern/regular expression an "in" parameterization of rule, wherein other regular expression instantiations are passed during instantiation yielding a production rule as a formula for building a regular expression (from other regular expressions) rather than the regular expression itself** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

Conclusion

13. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Andre R. Fowlkes whose telephone number is (571) 272-3697. The examiner can normally be reached on Monday - Friday, 8:00am-4:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571)272-3695. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Any inquiry of a general nature or relating to the status of this application should be directed to the **TC 2100 Group receptionist: 571-272-2100.**

ARF

TUAN DAM
SUPERVISORY PATENT EXAMINER